Desarrollo de una aplicación móvil para la detección de texto en imágenes TDetect

Macari Saura Lopez

Resumen — En la actualidad existe gran diversidad de aplicaciones móviles para abastecer las distintas necesidades de los usuarios, entre ellas están las que nos ayudan a localizar lugares de interés, reconocer canciones en tiempo real con solo escucharla, traducir textos en múltiples idiomas o los típicos y más comunes como son los buscadores web entre otros. Dentro de esta gran variedad aparecen algunas que nos ayudan al reconocimiento de caracteres o texto en imágenes. Actualmente existen algunos ejemplos como los OCRs que, a partir de una imagen capturada, son capaces de detectar el texto dentro de una imagen y convertirlo a un formato en concreto, o otras más interesantes como la recién adquirida por Google, Word Lens la cual está integrada en su aplicación Google Translate capaz de traducir texto en tiempo real con solo enfocar la cámara a la imagen a tratar. Este proyecto no es tan sofisticado como la de Google pero se podría decir que esta dentro de este grupo, el de reconocimiento de texto o caracteres a partir de una imagen, la finalidad es un realizar una aplicación en Adroid que mediante unas librerías OpenCV sea capaz de detectar texto dentro de las imágenes.

Palabras Clave — Android, aplicación móvil, reconocimiento de texto, imagen, OCR, REST, Cliente/Servidor,C++, OpenCV

Abstract — At present it exists a great variety of mobile applications to supply the different users' needs, among them are those that help us to locate places of interest, recognize songs in-real time just listening to them, translate texts in different languages or the typical and the most common web browsers among others. Within this range it appears some applications to help us to recognise the characters or text in images. Nowadays it exists as well some examples like the OCRs that, from a captured image, are able to detect text within an image and convert it to a particular format or others more interesting ones like the recently acquired by Google, Word Lens which is integrated in their application Google Translate and that is able to translate text in real-time by focusing the camera to the image to process. This project isn't as sophisticated as Google but it can be said that it is within that group, the text recognition or characters from an image. The aim is to carry out a full application to the Android through openCV libraries to allow to detect text within the pages.

Index Terms— Android, mobile aplications, text recognition, image, OCR, REST, Client/Server, C++, OpenCV

1 INTRODUCCIÓN

Hace unos años que los dispositivos móviles y las aplicaciones se hallan en nuestro día a día y están tan introducidos en nuestra Sociedad que parece que hayan existido durante siglos. Para muchos de nosotros, los Smartphone y el consumo de sus servicios se ha convertido en algo casi imprescindible.

En 2010, aproximadamente, fué el boom de los Smartphones y de las aplicaciones móviles. En ese mismo año, los smartphones y aplicaciones conquistaron nuestra sociedad. En 2011 hubo un crecimiento aproximado del 58% en la venta de estos dispositivos. Se vendieron aproximadamente unos 472 millones en todo el mundo y se lanzaban diariamente unes 698 aplicaciones compatibles.

Este fenómeno, en parte, es debido a que muchas de ellas nos ayudan con tareas del día a día como:

- Buscar lugares de interés.
- Conversar con otras personas
- Realizar retoques fotográficos.
- Encontrar y detectar el texto en imágenes.

El desarrollo de este proyecto se situaría dentro del grupo de encontrar y detectar texto en imágenes.

Se ha desarrollado una aplicación móvil la cual es capaz de reconocer texto dentro de una imagen.

El usuario, mediante una aplicación móvil, toma una fotografía y esta se almacena en el dispositivo. Con la aplicación puede mostrar las fotos almacenadas mediante una lista, borrar los elementos, hacer grande una foto y encontrar en que parte de la foto se encuentra el texto marcándolo mediante unos rectángulos.

Con esta versión, se ha diseñado una primera fase base y se podría ampliar posteriormente, implementando

- *E-mail de contacto:* macaxrc@gmail.com
- *Mención realizada: Ingeniería de software*
- *Trabajo autorizado por: Ernest Valveny (Departamento Ciencias de Computación)*
- *Curso 2014/15*

nuevas funcionalidades para darle mas potencia y hacer una herramienta más útil para el usuario final.

Algunas de las posibles funcionalidades que se podrian añadir podrían ser:

- Detectar exactamente que pone en dicho texto, ahora solo se busca y se reconoce el texto dentro de la imagen y se señaliza, pero no se detecta el texto introducido.

- Añadir la detección del idioma, saber si el texto encontrado està en catalán, castellano o ingles.

- Poder traducir el texto detectado.

- Realizar búsquedas en internet a partir del texto seleccionado.

1.1 Objetivo

El objetivo principal de este proyecto es realizar una aplicación móvil que sea capaz de detectar el texto dentro de una imagen. Como en la actualidad todos los Smartphones del momento disponen de una cámara, ésta se ha utilizado para capturar la imagen.

Un usuario con su teléfono móvil realiza una foto, la almacena en el dispositivo y mediante un proceso de computación se detecta el texto dentro de la imagen.

Imagen 1: Imagen antes del procesado

Imagen 2: Imagen después del procesado

En las imágenes anteriores se puede observar como quedaría una imagen antes y después del procesado.

La *Imagen 1* corresponde a una imagen tomada por la cámara del teléfono móvil y la *Imagen 2* la misma después del procesado. Como se puede apreciar, en la *Imagen 2* el resultado esperado es la misma imagen pero con la parte que contiene texto señalizado con unos rectángulos naranjas.

Desde la interfaz, el usuario podrá:

- Realizar una foto desde la aplicación (TDetect).
- Almacenar la foto en el dispositivo.
- Visualizar en una lista las fotos tomadas y las recibidas por el servidor
- Eliminar un elemento de la lista desde la lista
- Seleccionar la foto para verla en grande
- Eliminar una foto almacenada

1.2 Restricciones

En este proyecto existen varias restricciónes importantes.

El procesado de la imagen es un proceso muy costoso para un dispositivo móvil. Los smarphones actuales, aunque cada vez más, no son igual de potentes que los ordenadores convencionales y poseen varios hándicaps respecto a los ordenadores.

El hardware de un móvil es menor en cuanto a potencia y rendimiento, en canvio, un ordenador de sobremesa goza de mayor potencia para procesar datos. Además, los teléfonos requieren de una alimentación externa debido a que no están conectados a una red eléctrica para que fúncionen. Precisan una batería la cual tiene un tiempo limitado y su duración dependerá de su uso.

Si el procesado de la imagen se ejecutara directamente en el teléfono podría ocurrir lo siguiente:

- Que el procesado no se pudiera llevar a cabo porque el hardware de los smartphones no son lo suficientemente potentes para soportar el cálculo.

- El consumo energético aumentaría considerablemente y con ello una disminución drástica de la batería.

- La temperatura del dispositivo se vería afectada.

Por estas razones, esta parte tiene que ser tratada por un dispositivo externo.

Lo que se propone, es desarrollar una interfaz para Android (Cliente), mediante la cual, el usuario pueda enviar la imagen para el procesado a un ordenador (Servidor), éste trate la imagen y devuelva la imagen al cliente.

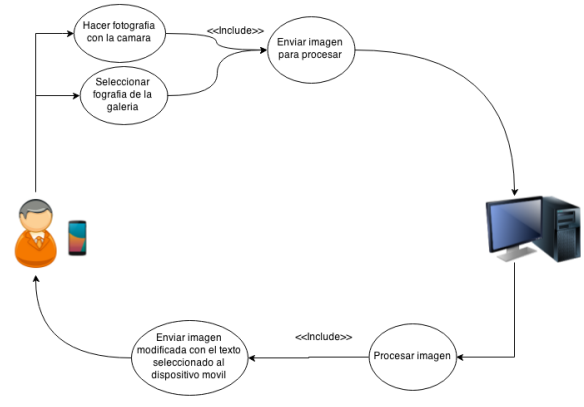


Imagen 3: Diagrama caso de uso del funcionamiento

1.3 Estado del arte

Para el inicio del proyecto se ha proporcionado por parte del Centro de Visión por computador (CVC) de la universidad autónoma de Barcelona, la parte encargada del tratado de la imagen.

Se parte de un proyecto realizado en lenguaje *C++* que, mediante unas libreriras *Open CV* y recibiendo una imagen almacenada en disco, es capaz de tratar la imagen.

Open CV son unas librerías Open Source de visión artifical y machine learning que poseen una comunidad de más de 47000 personas y más de 7 millones de descargas.

En estas librerías se aprecian más de 2500 algoritmos que nos permiten, entre otras cosas, identificar objetos, caras, clasificaciones humanas, encontrar similitud entre imágenes, seguir el movimiento de los ojos o detectar texto en imágenes.

2 METODOLOGIA

Para un desarrollo de software ágil existe gran variedad de métodos, los cuales, hacen que llegar al objetivo deseado sea todo un éxito.

La mayoría de ellos son utilizados por equipos de desarrollo de varias personas. En este caso, solamente existe el profesor, el alumno y los objetivos claramente marcados y no cambiantes ya que desde el inicio del proyecto se establecieron.

Se podría decir que en este proyecto se ha utilizado un modelo en Espiral, donde sobre cada fase o iteración del proyecto se han evaluado diferentes aspectos.

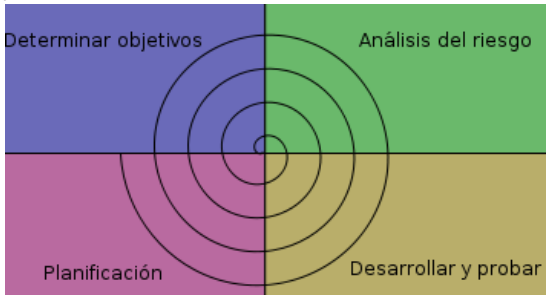


Imagen 4: Fases del desarrollo en Espiral

El Desarrollo en Espiral, tiene en cuenta 4 tareas importantes:

1- Determinar objetivos
2- Analisis de riesgos
3- Desarrollar y probar
4- Planificación

Estas tareas se van aplicando durante las diferentes fases del proyecto. En cada fase se definían los objetivos de esa tarea, los posibles riesgos, ver si era viable o no desarrollar esa parte, desarrollar la parte definida y planificar la siguiente interación

A continuación se puede observar todas las fases del proyecto.

FASES PROYECTO
Fase 1: *Ejecutar y probar la decodificación de la imagen que se proporcionó.*
Fase 2: *Buscar información y analizar las distintas formas de llevar a cavo una comunicación cliente servidor con un dispositivo móvil.*
Fase 3: *Analizar los distintos servidores de aplicaciones y escoger el que más se adapte al proyecto.*
Fase 4: *Diseñar en papel una interfaz de usuario y enseñarlo a varios usuarios para conseguir una buena experiencia de uso.*
Fase 5: *Desarrollar la parte cliente y testearla.*
Fase 6: *Adaptar las 2 partes (cliente/servidor) para realizar la conexión y conseguir el envío y la decodificación de la imagen.*
Fase 7: *Entrega final y presentación.*

Tabla 1: Fases del proyecto

Algunas ventajas del modelo en espiral son:

- Reduce el riesgo del proyecto: Al establecerse los objetivos al inicio de cada fase y evaluar su implementación hace que el riesgo sea menor, al principio de cada iteración ya se sabe si el objetivo es o no viable.
- Incorpora objetivos de calidad: El modelo en espiral dice que los objetivos tiene que estar claramente marcados al inicio de cada iteración. En el proyecto dichos objetivos se maracaron al inicio y no fueron alterados.

Una desventaja sobre este modelo es:

- Requiere habilidad para la evaluación del riesgo: Si no se hace un análisis exausto de como se quiere implementar puede haber problemas en el desarrollo.

3 PLANIFICACIÓN

Como se ha comentado anteriormente, el proyecto se desglosó en 7 partes y se realizó una estimación aproximada en horas.

Fases	*Horas (Aprox)*
Fase 1	*10h*
Fase 2	*20 h*
Fase 3	*80h*
Fase 4	*50h*
Fase 5	*70h*
Fase 6	*70h*
Fase 7	*30h*
TOTAL	*330h*

Tabla 2: Horas del proyecto

En la tabla anterior se puede visualizar la planificación incial. Ésta sufrió algún cambio en el orden de ejecución pasando a ejecutarse en el siguiente orden:

1. Fase 1
2. Fase 4
3. Fase 5
4. Fase 2
5. Fase 3
6. Fase 6
7. Fase 7

Esto fue debido a que se decidió primero empezar a implementar la interfaz para, posteriormente, desarrollar la comunicación. No obstante, este cambio no afectó negativamente en objetivo establecido.

4 DESARROLLO

En este apartado se describirán los pasos más importantes para lograr el objetivo del proyecto.

4.1 INTERFAZ GRAFICA

Para el desarrollo de la interfaz, antes de empezar a implementarla, inicialmente, se diseñaron varios mockups en papel para conseguir feedback de algunos usuarios y posteriormente poder mejorarla antes de empezar a desarrollarla.

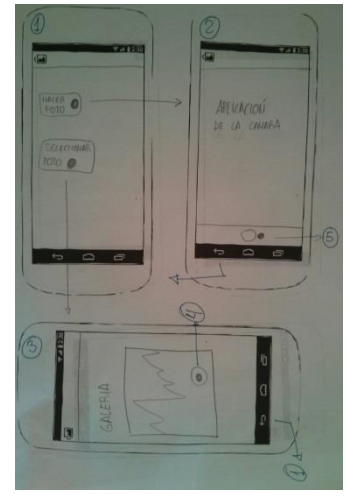


Imagen 5: Primera versión de la interfaz

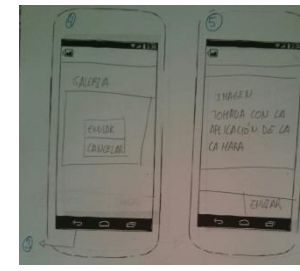


Imagen 6: Primera versión de la interfaz

El primer boceto era muy sencillo. Después de estudiarlo y valorar diferentes opiones de possibles usuarios, se decidió investigar para ver como eran los estilos de diseño que utilizaban las últimas versiones de android 4.x.x.

Muchas aplicaciones basadas en esta versión de Android como Gmail o Hangouts ya implementan los menús laterales (Naviagation Drawer).

Imagen 7: Lista de mensajes de Hangouts

Imagen 8: Menu lateral de la aplicacón Hangouts

Como el boceto inicial no acabó de gustar a los usuarios finales y tampoco se adaptaba a los patrones de diseño de interfaces que propone Android, se volvió a realizar otro mockup que se asemejara más a estos nuevos diseños. La idea de confeccionar un buen diseño de interfaces es que el usuario sin leerse un manual de instrucciones entienda como funciona la aplicación y que tenga una buena usabilidad.

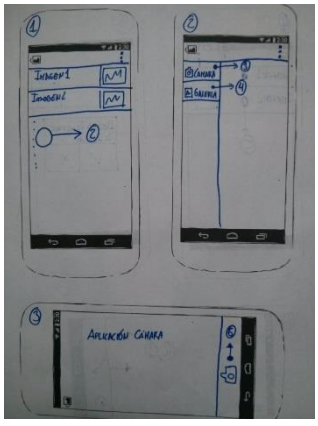


Imagen 9: Segundo boceto de la interficie

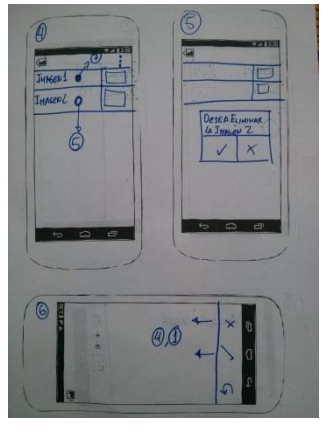


Imagen 10: Segundo boceto de la interficie

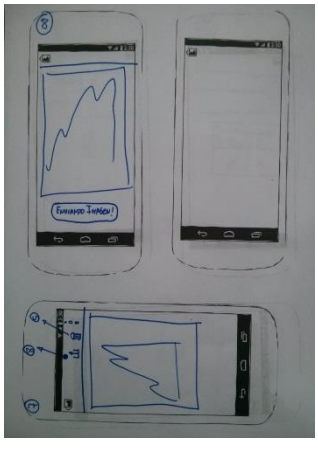


Imagen 11: Segundo boceto de la interficie

En las imágenes 9,10 y 11 se puede observar la interfaz definitiva, intentando que se parezca más a lo estándar. En la imagen 9 figura 1, se puede observar que si hacemos swipe de izquierda a derecha aparece un menú con 2 opciones (cámara, galeria).
Si seleccionamos "galeria" se nos abrirá otra pantalla con las imágenes caputradas con la aplicación (ListView) desde la que podremos seleccionar una foto haciendo click (Imagen 10, figura 4). De esta manera se nos ampliará y nos aparecerá arriba a la derecha (action bar) una opción para enviar la foto a analizar o borrar (Imagen 14, figura 7).
Si en cambio seleccionamos hacer foto, se nos abrirá la aplicación de la cámara para realizar la foto (Imagen 10 figura3).

Una vez decidida la interfaz gráfica se empezó a implementar.

La interfaz esta dividida en 4 partes claramente diferenciadas. A continuación, se definirán las distintas partes y a que clase correspondería en cada apartado:

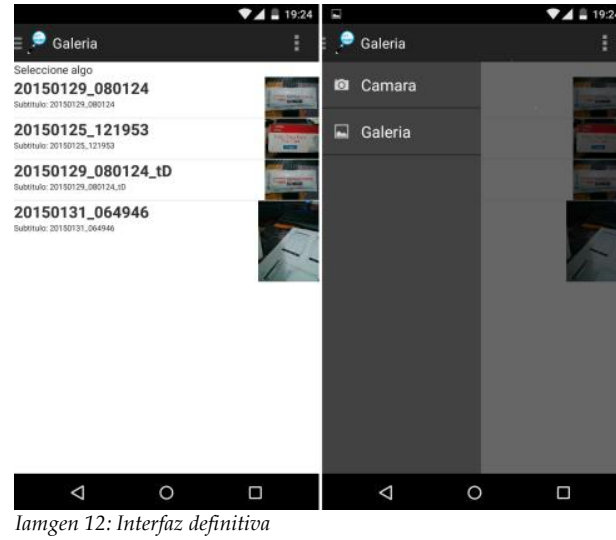
- Gestiónar del menú lateral:
  - MainActivity.java
  - Item_objt.java
  - NavigationAdapter.java

- Mostrar fotos almacenadas
  - GaleriaFragment.java
  - FotoRecord.java
  - GaleriaAdapter.java

- Enviar las fotos al servidor
  - UploaderFoto.java

- Hacer grande la foto
  - BigSizeImageView.java

La interfaz ha quedado de la siguiente manera:



Iamgen 12: Interfaz definitiva

4.2 COMUNICACIÓN CON EL SERVIDOR

A continuación, se explicará como se ha desarrollado la comunicación cliente servidor.

Se planteaba un problema importante. La parte encargada de tratar la imagen estaba desarrollada en C++ y la interfaz del cliente en Android (Java). El principal problema entre ambos, es que claramente son lenguajes diferentes, con lo cual, había que encontrar alguna manera para que la comunicación fuera completamente independiente del lenguaje utilizado.

Para solventarlo, se determinó realizar una arquitectura REST en la comunicación.

REST es una arquitectura basada en HTTP la cual nos ayuda a producir una comunicación sin estado, es decir, ni el cliente ni el servidor necesita saber el estado para la comunicación. El cliente, mediante una conexión HTTP, efectua una petición por medio de POST, GET, PUT y DELETE a un puerto concreto y el servidor ya es capaz de recibir la petición y devolver la respuesta. En cada mensaje HTTP yace la información necesaria para comprender la petición.

Para adaptar REST al proyecto inicialente habían 2 opciones:

- ASP.NET MVC 3

  Las ventajas que tenia esta primera opción eran que era fácilmente integrable en Visual estudio y que poseia gran información. Para poder utilizarlo en contra solo funcionaba con C#, con lo cual se desacarto.

- C++ Rest SDK

La segunda opción C++ Rest SDK eran otras librerías que eran fáciles de instalar en nuestro proyecto y que se lograria adaptar en la parte de detección de texto ya que se encontraban en C++, era la candidata para solventar el problema de la comunicación, pero se acabó descartando por falta de documentación, era escasa haciendo que si se optava por esta solución no se alcanzaría al objetivo establecido ya que el coste y tiempo se verían incrementados.

Por estos motivos, se decidió realizar un WorkAround y hallar una alternativa optima para solventar el problema.

La solución es una comunicación Rest mediante PHP.

Ventajas:

- PHP es un lenguaje muy empleado en comunicaciónes web. Es un lenguaje que se ejecuta en la parte servidora de la arquitectura.
- Hay muchas aplicaciones desarrolladas en PHP con lo cual abunda documentación como ejemplos posibles.

Inconvenentes:
- Se utiliza un lenguaje intermedio para las llamadas al procesado de la imagen, puede afectar al rendimiento.
- Tiene limites de memória (se pueden solucionar) para ejecutar procesos.

Por un lado tenemos la comunicación en si, Cliente/Servidor (Android-PHP) y por otro la comunicación con la parte encargada (PHP-C++) del tratado de la imagen.

Es decir, el cliente envia una petición POST mediante HTTP del tipo http:\\ipServidora:puerto\NombreRecurso.php, el servidor recoge la petición, éste hace una llamada a un proceso escrito en C++ y cuando el proceso finaliza, el código PHP retorna una respuesta al cliente en formato JSON.

En la siguiente imagen se puede ver el proceso de comunicación.

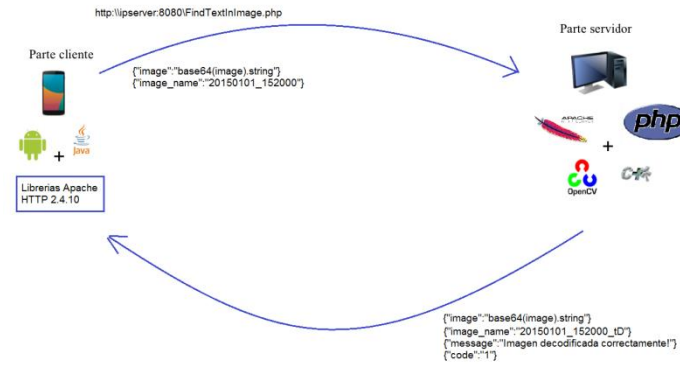


Imagen 13: Comunicación Cliente/Servidor

4.2.1 Configuración del servidor

PHP es un lenguaje interpretado que se ejecuta en la parte servidor. Para lograr utilizarlo, es necesario instalar las librerías de php y disponer de un servidor de aplicaciones. Como servidor de aplicación se ha apostado por Apache en su versión 2.2 ya que es un servidor de aplicación bastante utilizado y con mucha información disponible.

Apache es el encargado de levantar un servicio en Windows para que esté escuchando continuamente y con la ayuda del código en PHP sean los encargados de recibir peticiones HTTP.

Para que funcione correctamente se tiene que realizar varias configuraciones para lograr ofrecer al cliente las peticiones que necesita.

4.2.2 Envío y recepción de la imagen

Para la comunicación mediante HTTP han sido necesarias unas librerías de Apache, httpmime-4.2.4.jar. Estas librerías se han tenido que integrar en la parte cliente del proyecto. Asimismo, se ha tenido que dar permiso a nuestra aplicación para que pueda conectarse a internet y para ello se ha modificado el archivo Andoroid-manifest.xml

añadiendo la siguiente línea.

```
<uses-permission android:name="android.permission.INTERNET"/>
```

Problema:

Las imágenes no se enviaban ocasionalemente debido a su tamaño o a la calidad de la fotografia.

Solución:

Convertir la imagen en un string en base64. Este método es muy utilizado para enviar información a través de la red. Grácias a esta codificación nos aseguramos de que no haya pérdida de información en el envío. No entramos en investigar como funciona internamente la codificación en base64 ni porque en imágenes grandes había este problema ya que, enviando los datos de esta forma, funcionó todo sin problemas.

Estructura de la información de envio:

Los datos se envían en formato JSON. Es una representación de la información que quiere substituir a los XML tradicionales. Su formato se representa mediante clave/valor y lo ventajoso es que se puede leer desde cualquier lenguaje de programación.

Su formato es el siguiente:

```
{"nombre":"Antonio Probencio"}
```

En el caso de la imagen, lo que se envía, es el nombre, la foto y la foto en formato base64.

Por ejemplo:

```
{"nombreImagen":"imagen1.jpg","imagen":"4AAQSkZJRgABAQAAAQABAAD\/2wBDAAIBAQEBAQIBAQECAgICAgQDA"}
```

Los datos, al enviarlos codificados entre cliente y servidor, estos son decodificados en ambos extremos.

Para poder seguir la ejecución, se generan unos archivos de log en formato.txt para poder trazar la ejecución.

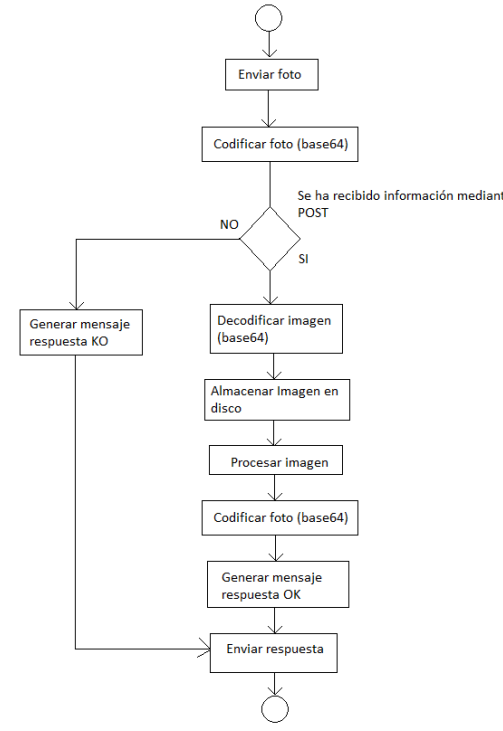


Imagen 14: Flujo de la comunicación

4.2.3 Tratado de la imagen

En este apartado se elaborarán breves comentarios sobre la decodificación de la imagen ya que el código ha sido proporcionado por el tutor del proyecto.

Para poder ejecutar este código desde PHP y que recibiera una imagen como parámetro, se tuvo que adapatar un poco.

Inicialmente el código después de la ejecución, mostraba el resultado cargado en memoria, es decir, aparecia la foto en la pantalla. Para cubrir las necesidades del proyecto se tuvo que:

- Tomar el primer argumento de la llamada para la decodificación.
- Hacer que la imagen se guardara en disco.
- Renombrar la imagen una vez realizada la ejecución
- Crear archivos .txt de log para poder trazar la ejecución.
- Compilar el código y colocar el .exe en una carpeta del apache.

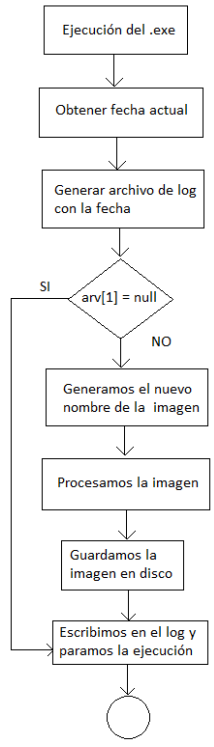


Imagen 15: Diagrama de flujo del procesado de la imagen

Problema

El tratado de la imagen como se ha comentado al principio el documento es un proceso bastante costoso Después de varias ejecuciones, se ha dado el caso que según el tipo de imagen, ya sea porque tiene demasido texto o colores muy similares, puede llegar a consumir mucha memoria entre 1GB y 2Gb y el tiempo de tratado aumentar considerablemente. En estos casos la aplicación sufria un desbordamineto de memoria ya que la llamada de php tiene un límite.

Solución

Para solventarlo se tuvo que asignar más memoria al principio del PHP y darle mas tiempo de procesado.

```
ini_set('memory_limit', '512M');
ini_set('max_execution_time', 1200);
```

5 Requisitos

El interfaz de usuario para que funcione correctamente en el dispositivo, éste tiene que tener el SDK 11 (Android 3.0).

Las pruebas realizadas han sido con un Nexus 4 Android 5.01 (Lolipop).

En la parte servidora Windows 7 ultimate con Intel i7 950 con 6GB de Ram. Apache 2.2 y PHP 5.3.

7 Conclusiones

Como se puede observar en la distintas fotografias el resultado obtenido es el esperado. Se ha conseguido realizar una comunicación cliente servidor y tratar la imagen para la detección de texto, es decir, que el objetivo incial del proyecto se ha logrado.

Por otra parte, si que es cierto que hubiese estado mas interesante si se hubiesen introducido mas funcionalidades, como añadir la parte de detección de texto literal i poder detectar el idioma o traducir el texto, pero por falta de tiempo no ha sido posible. Al inicio del documento se expone que es una versión base y que a partir de aquí se podrá ir añadiendo más opciones.

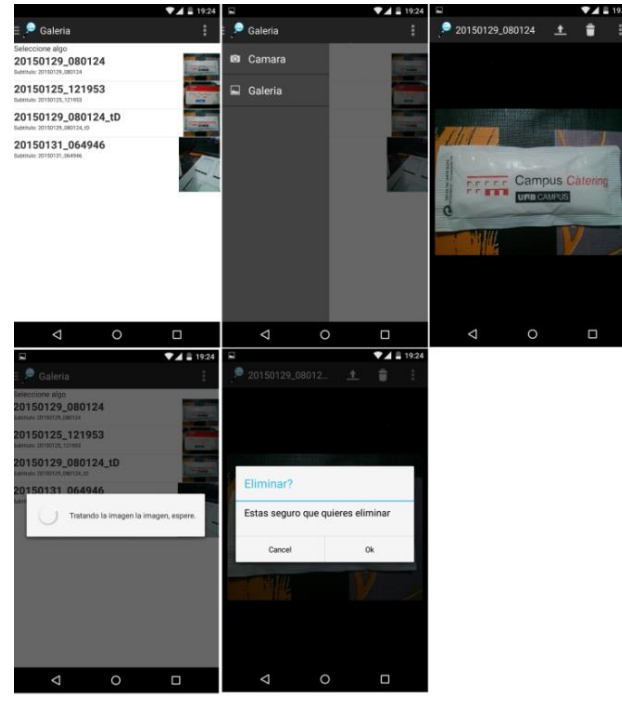


Imagen 16: Aplicación acabada

Imagen 17: Foto antes de procesar

Imagen 18: Foto después de procesar

Imagen 19: Foto antes de procesar

Imagen 20: Foto después de procesar

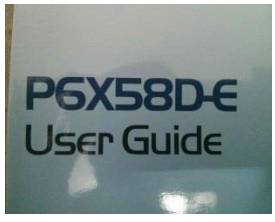


Imagen 21: Foto antes de procesar

Imagen 22: Foto después de procesar

Como se puede apreciar en la *Imagen 18*, hay parte de la fotografia que se ha reconocido el texto como es el caso de "Eroski" y "Basic". Hay que mencionar que el reconocimiento de texto a veces no detecta el texto correctamente.

8 Agradecimientos

A mí novia Ana Espinosa por aguantarme durante todo este tiempo, ya no en el desarrollo del proyecto, sinó durante toda la carrera.

A mis padres por el apoyo recibido.

A mí cuñado M.Antonio Espinosa por darme ánimos en los momentos de tensión.

A Eloi Ballarà Madrid e Isaac Serrano por compartir esos momentos de desespero y gloria durante el transcurso de todo el proyecto.

A mis amigos de toda la vida, que aunque no saben ni lo que he estado haciendo en estos últimos años de mi vida, han estado ahí.

9 Bibliografia

opencv.org: Web oficial de las librerías de visión artificial.

www.sgoliver.net: Blog con tutoriales para aprender a programar en android.

apache.org: Web oficial de Apache configuración e instalación.

picarcodigo.blogspot.com.es/2014/05/webservice-subir-imagen-servidor-desde: Blog con ejemplo de envio de imágenes en Android.

developer.android.com: Web oficial desarrollo Android

casablanca.codeplex.com: Web de las librerías C++ Rest SDK.

stackoverflow.com: Foro con mucha información sobre programación en general.

forosdelweb.com: Foros de discusión de proramación en general.

php.net: Documentación general de php, versiones, historia, funciones disponibles